A

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

| APPLICATION NO. | FILING DATE | FIRST NAMED INVENTOR | ATTORNEY DOCKET NO. | CONFIRMATION NO. |
|---|---|---|---|---|
| 09/667,430 | 09/21/2000 | Larry Koved | YOR9-2000-0253 (728-170) | 8852 |

7590 01/14/2004

Paul J Farrell Esq
Dilworth & Barrese
333 Earle Ovington Boulevard
Uniondale, NY 11553

| EXAMINER |
|---|
| ROCHE, TRENTON J |

| ART UNIT | PAPER NUMBER |
|---|---|
| 2124 | 3 |

DATE MAILED: 01/14/2004

Please find below and/or attached an Office communication concerning this application or proceeding.

| | | |
|---|---|---|
| *Office Action Summary* | **Application N .** 09/667,430 | **Applicant(s)** KOVED ET AL. |
| | **Examiner** Trent J Roche | **Art Unit** 2124 |

***-- The MAILING DATE of this communication appears on the c ver sheet with the c rrespondence address --***

**Period for Reply**

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133).
- Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

**Status**

1)☒ Responsive to communication(s) filed on 21 September 2000.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

**Disposition of Claims**

4)☒ Claim(s) 1-71 is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) 1-71 is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

**Application Papers**

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on 21 September 2000 is/are: a)☐ accepted or b)☒ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

**Priority under 35 U.S.C. §§ 119 and 120**

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

13)☐ Acknowledgment is made of a claim for domestic priority under 35 U.S.C. § 119(e) (to a provisional application) since a specific reference was included in the first sentence of the specification or in an Application Data Sheet. 37 CFR 1.78.

a) ☐ The translation of the foreign language provisional application has been received.

14)☐ Acknowledgment is made of a claim for domestic priority under 35 U.S.C. §§ 120 and/or 121 since a specific reference was included in the first sentence of the specification or in an Application Data Sheet. 37 CFR 1.78.

**Attachment(s)**

| | |
|---|---|
| 1) ☒ Notice of References Cited (PTO-892) | 4) ☐ Interview Summary (PTO-413) Paper No(s). _____ . |
| 2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948) | 5) ☐ Notice of Informal Patent Application (PTO-152) |
| 3) ☒ Information Disclosure Statement(s) (PTO-1449) Paper No(s) 2 . | 6) ☐ Other: . |

## DETAILED ACTION

1. Claims 1-71 have been examined.

### *Drawings*

2. The drawings are objected to because Fig. 1 shows an element titled "JAN" which is not disclosed in the related section of the specification. It is unclear as to what "JAN" is referring to. Further, it is suggested that reference numbers be added to the drawings and to the related sections of the specification for added clarity (Note 37 CFR 1.74). A proposed drawing correction or corrected drawings are required in reply to the Office action to avoid abandonment of the application. The objection to the drawings will not be held in abeyance.

### *Claim Rejections - 35 USC § 112*

3. The following is a quotation of the second paragraph of 35 U.S.C. 112:

> The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

4. Claims 4, 11, 18, 23, 26-37, 41, 48 and 57 are rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention.

In claim 26 is it unclear as to whether attention is directed to a device comprising a variety of items or to a component comprising a variety of items. Note that the claims states "A device for detecting…in a program component, said component being written in an object-oriented language,

**comprising**..." It is noted that the wording of the claim makes it indefinite as to which element, the device or the component, the word comprising is directed to.

Consequently, claims 27-37 are rejected for being dependent on claim 26. For purposes of examination, the claim will be interpreted to read "A device for detecting mutability of variables, objects, fields, and classes in a program component, said component being written in an object-oriented programming language, said device comprising..."

5. Claims 4, 11, 18, 23, 41, 48 and 57 contain the trademark/trade name Java. Where a trademark or trade name is used in a claim as a limitation to identify or describe a particular material or product, the claim does not comply with the requirements of 35 U.S.C. 112, second paragraph. See *Ex parte Simpson*, 218 USPQ 1020 (Bd. App. 1982). The claim scope is uncertain since the trademark or trade name cannot be used properly to identify any particular material or product. A trademark or trade name is used to identify a source of goods, and not the goods themselves. Thus, a trademark or trade name does not identify or describe the goods associated with the trademark or trade name. In the present case, the trademark/trade name is used to identify/describe the Java programming language by Sun Microsystems and, accordingly, the identification/description is indefinite.

## *Claim Rejections - 35 USC § 101*

6. 35 U.S.C. 101 reads as follows:

Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.

Claims 26-37 are rejected under 35 U.S.C. 101 because the claimed invention is directed to non-statutory subject matter.

The invention as disclosed in claims 26-37 is directed to non-statutory subject matter. The claimed invention as a whole must accomplish a practical application. That is, it must produce a "useful, concrete and **tangible** result." (State Street Bank & Trust Co. v. Signature Financial Group Inc., 149 F.3d at 1373, 47 USPQ2d at 1601-02.)

Specifically, the claims are directed to a device comprising a core library, a data-flow analysis engine and a utility module. These elements are interpreted to be software based, and is thus not necessarily embodied in a tangible piece of hardware such as a computer system. Thus, Applicants fail to disclose that the language is tangibly embodied and executed by a piece of hardware and that their functions have practical applications which produce useful, concrete, and tangible results under the State Street Formulation.

On this basis, claims 26-37 are rejected under 35 U.S.C. § 101.

## *Claim Rejections - 35 USC § 102*

7. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

A person shall be entitled to a patent unless –

(a) the invention was known or used by others in this country, or patented or described in a printed publication in this or a foreign country, before the invention thereof by the applicant for a patent.

8. Claims 1-71 are rejected under 35 U.S.C. 102(e) as being anticipated by U.S. Patent 6,085,035 to Ungar.

**Regarding claim 1:**

Ungar teaches:

- a method of detecting mutability of variables, objects, fields, and classes in a program component ("The invention detects whether the variable is type immutable..." in col. 7 lines 30-31)
- said component being written in an object oriented programming language ("Another embodiment optimizes OOP methods..." in col. 8 lines 49-50)
- determining whether any variable in the program component could undergo a state modification of a first type, said first type state modification being made by at least one method that is within the program component ("if the stored data-values are of only one type the data-values are type immutable" in col. 8 lines 34-35)
- performing encapsulation analysis to determine whether any variable in the program component could undergo a state modification of a second type, said second type state modification being made by at least one method that is not within the program component ("the 'determine type usage pattern' procedure determines whether only data-values having a specific type are stored in the variable" in col. 8 lines 30-32)
- wherein a variable is mutable if its state ever changes after said variable is initialized, the state of said variable being its value together with a state of any referenced object ("Mutable type variables store data-values of different types" in col. 5 lines 8-9)
- wherein an object is mutable if its state ever changes after said object is initialized, said state of said object being a set of states of all associated variables ("Mutable type variables store data-values of different types" in col. 5 lines 8-9. Further, "the 'z instance variable' storage contains the instance variables for the 'z object' structure" in col. 7 lines 7-8)

- wherein a field is mutable if any variable corresponding to said field is mutable ("Mutable type variables store data-values of different types" in col. 5 lines 8-9. Further, fields are inherently an object of the system.)
- wherein a class is mutable if any instance fields implemented by said class are mutable ("Mutable type variables store data-values of different types" in col. 5 lines 8-9. Further, A 'z object' structure contains information that is specific to each instantiation of an object of the 'z' class/map" in col. 7 lines 1-3)

as claimed.

**Regarding claim 2:**

The rejection of claim 1 is incorporated, and further, Ungar discloses detecting possible first type state modifications as claimed ("The invention detects whether the variable is type immutable, whether the data-values stored in the variable have a preferred type…" in col. 7 lines 30-32)

**Regarding claim 3:**

The rejection of claim 1 is incorporated, and further, Ungar discloses an encapsulation analysis step as claimed ("the 'determine type usage pattern' procedure determines whether only data-values having a specific type are stored in the variable" in col. 8 lines 30-32. Further, note col. 8 lines 35-46)

**Regarding claim 4:**

The rejection of claim 1 is incorporated, and further, Ungar discloses a method being implemented in a Java environment as claimed (Note page 2, Table 1)

**Regarding claim 5:**

The rejection of claim 1 is incorporated, and further, Ungar discloses identifying isolation faults as claimed ("evaluates the type mutability of the typed data-values stored in the variable...that is, the 'determine type usage pattern' procedure determines whether only data-values having a specific type are stored in the variable" in col. 8 lines 29-32)

**Regarding claim 6:**

The rejection of claim 1 is incorporated, and further, Ungar discloses identifying fields and objects that can be determined to be constants as claimed ("The invention detects variables that have immutable types (from the 'determine type usage pattern' procedure)..." in col. 8 lines 61-62. The objects are determined to be immutable and therefore constant as they are not detected as mutable.)

**Regarding claim 7:**

Ungar teaches:

- a method of detecting mutability of classes in a program component ("The invention detects whether the variable is type immutable..." in col. 7 lines 30-31. Further, A 'z object' structure contains information that is specific to each instantiation of an object of the 'z' class/map" in col. 7 lines 1-3)
- said component being written in an object oriented programming language ("Another embodiment optimizes OOP methods..." in col. 8 lines 49-50)
- obtaining a set of classes, each of said classes being classified as one of mutable, immutable, and undecided ("The first case is whether data-values stored in a variable all have the same

type...the second case is whether the variable has a preferred type...the third case is when the variable does not have a preferred type" in col. 7 lines 33-47)

- testing each undecided class (Note Fig. 3 and the corresponding section of the disclosure)
- testing each field in said undecided class being tested (Note Fig. 3 and the corresponding section of the disclosure)
- determining whether any variable in the program component could undergo a state modification of a first type, said first type state modification being made by at least one method that is within said component ("if the stored data-values are of only one type the data-values are type immutable" in col. 8 lines 34-35)
- performing encapsulation analysis to determine whether any variable corresponding to said each field could undergo a state modification of a second type, said second type state modification being made by at least one method that is not within said component ("the 'determine type usage pattern' procedure determines whether only data-values having a specific type are stored in the variable" in col. 8 lines 30-32)
- classifying each field as immutable if no possible first type of second type state modifications are found ("if the data-values are of only one type the data-values are type immutable" in col. 8 lines 34-35)
- classifying said each field as undecided if there is insufficient class mutability information ("The third case is when the variable does not have a preferred type" in col. 7 lines 46-47)
- classifying said each field as mutable ("The typed data-values are type mutable if data-values of different types can be stored in the variable" in col. 8 lines 32-34)
- re-classifying said undecided class as mutable (Note Fig. 4A and the corresponding section of the disclosure)

- re-classifying said undecided class as immutable (Note Fig. 4A and the corresponding section of the disclosure)
- repeating said testing each undecided class step until a number of undecided classes after a repetition of said testing step is identical to a number of undecided classes before the repetition of said testing step (Note Fig. 3 and the corresponding section of the disclosure. The invention would repeat the classification procedure until all objects are classified.)
- re-classifying remaining undecided classes as mutable classes (Note Fig. 4A and the corresponding section of the disclosure)

as claimed.

**Regarding claim 8:**

Ungar teaches:

- a method of detecting mutability of classes in a program component ("The invention detects whether the variable is type immutable..." in col. 7 lines 30-31. Further, A 'z object' structure contains information that is specific to each instantiation of an object of the 'z' class/map" in col. 7 lines 1-3)
- said component being written in an object oriented programming language ("Another embodiment optimizes OOP methods..." in col. 8 lines 49-50)
- obtaining a set of classes, each of said classes being classified as one of mutable, immutable, and undecided ("The first case is whether data-values stored in a variable all have the same type...the second case is whether the variable has a preferred type...the third case is when the variable does not have a preferred type" in col. 7 lines 33-47)
- testing each undecided class (Note Fig. 3 and the corresponding section of the disclosure)

- testing each instance field in said undecided class being tested (Note Fig. 3 and the corresponding section of the disclosure)
- determining whether any variable in the program component could undergo a state modification of a first type, said first type state modification being made by at least one method that is within said component ("if the stored data-values are of only one type the data-values are type immutable" in col. 8 lines 34-35)
- performing encapsulation analysis to determine whether any variable corresponding to said each instance field could undergo a state modification of a second type, said second type state modification being made by at least one method that is not within said component ("the 'determine type usage pattern' procedure determines whether only data-values having a specific type are stored in the variable" in col. 8 lines 30-32)
- classifying said each instance field as immutable if no possible first type of second type state modifications are found ("if the data-values are of only one type the data-values are type immutable" in col. 8 lines 34-35)
- classifying said each instance field as undecided if there is insufficient class mutability information ("The third case is when the variable does not have a preferred type" in col. 7 lines 46-47)
- classifying said each instance field as mutable ("The typed data-values are type mutable if data-values of different types can be stored in the variable" in col. 8 lines 32-34)
- re-classifying said undecided class as mutable (Note Fig. 4A and the corresponding section of the disclosure)
- re-classifying said undecided class as immutable (Note Fig. 4A and the corresponding section of the disclosure)

- repeating said testing each undecided class step until a number of undecided classes after a repetition of said testing step is identical to a number of undecided classes before the repetition of said testing step (Note Fig. 3 and the corresponding section of the disclosure. The invention would repeat the classification procedure until all objects are classified.)
- re-classifying remaining undecided classes as mutable classes (Note Fig. 4A and the corresponding section of the disclosure)

as claimed.

**Regarding claim 9:**

The rejection of claim 8 is incorporated, and further, Ungar discloses detecting possible first type state modification of a value as claimed ("if the stored data-values are of only one type the data-values are type immutable" in col. 8 lines 34-35)

**Regarding claim 10:**

The rejection of claim 8 is incorporated, and further, Ungar discloses detecting possible second type modification of a value as claimed ("the 'determine type usage pattern' procedure determines whether only data-values having a specific type are stored in the variable" in col. 8 lines 30-32)

**Regarding claim 11:**

The rejection of claim 8 is incorporated, and further, Ungar discloses a method being implemented in a Java environment as claimed (Note rejection regarding claim 4)

**Regarding claim 12:**

The rejection of claim 8 is incorporated, and further, Ungar discloses identifying fields and objects as constants as claimed (Note rejection regarding claim 6)

**Regarding claim 13:**

The rejection of claim 8 is incorporated, and further, Ungar discloses testing mutability of each undecided class as claimed (Note rejection regarding claim 8)

**Regarding claim 14:**

The rejection of claim 8 is incorporated, and further, Ungar discloses identifying isolation faults as claimed (Note rejection regarding claim 5)

**Regarding claim 15:**

The rejection of claim 8 is incorporated, and further, Ungar discloses the steps of testing mutability of undecided class fields as claimed (Note rejection regarding claim 8)

**Regarding claim 16:**

The rejection of claim 15 is incorporated, and further, Ungar discloses the steps of testing mutability of undecided class fields as claimed (Note rejection regarding claim 9)

**Regarding claim 17:**

The rejection of claim 15 is incorporated, and further, Ungar discloses performing encapsulation analysis as claimed (Note rejection regarding claim 3)

**Regarding claim 18:**

The rejection of claim 13 is incorporated, and further, Ungar discloses a method being implemented in a Java environment as claimed (Note rejection regarding claim 4)

**Regarding claim 19:**

Ungar teaches:

- a method of detecting mutability of classes and class variables in a program component ("The invention detects whether the variable is type immutable..." in col. 7 lines 30-31. Further, A 'z object' structure contains information that is specific to each instantiation of an object of the 'z' class/map" in col. 7 lines 1-3)
- said component being written in an object oriented programming language ("Another embodiment optimizes OOP methods..." in col. 8 lines 49-50)
- obtaining a set of classes, each of said classes being classified as one of mutable, immutable, and undecided ("The first case is whether data-values stored in a variable all have the same type...the second case is whether the variable has a preferred type...the third case is when the variable does not have a preferred type" in col. 7 lines 33-47)
- testing each undecided class (Note Fig. 3 and the corresponding section of the disclosure)
- testing mutability of each instance field in said undecided class being tested (Note Fig. 3 and the corresponding section of the disclosure)
- classifying an instance field as immutable if no possible state or encapsulation analysis modifications are found ("if the data-values are of only one type the data-values are type immutable" in col. 8 lines 34-35)

- classifying an instance field as undecided if there is insufficient class mutability information ("The third case is when the variable does not have a preferred type" in col. 7 lines 46-47)
- classifying an instance field as mutable ("The typed data-values are type mutable if data-values of different types can be stored in the variable" in col. 8 lines 32-34)
- re-classifying said undecided class as mutable (Note Fig. 4A and the corresponding section of the disclosure)
- re-classifying said undecided class as immutable (Note Fig. 4A and the corresponding section of the disclosure)
- repeating said testing each undecided class step until a number of undecided classes after a repetition of said testing step is identical to a number of undecided classes before the repetition of said testing step (Note Fig. 3 and the corresponding section of the disclosure. The invention would repeat the classification procedure until all objects are classified.)
- re-classifying remaining undecided classes as mutable classes (Note Fig. 4A and the corresponding section of the disclosure)
- testing mutability of each class field in each class ("a 'determine type usage pattern' procedure determines whether only data-values having a specific type are stored...the typed data-values are type mutable if data-values of different types can be stored..." in col. 8 lines 30-34)

as claimed.

**Regarding claim 20:**

The rejection of claim 19 is incorporated, and further, Ungar discloses testing mutability of a field as claimed. (Note rejection regarding claim 7)

**Regarding claim 21:**

The rejection of claim 20 is incorporated, and further, Ungar discloses a first type state modification as claimed. (Note rejection regarding claim 9)

**Regarding claim 22:**

The rejection of claim 20 is incorporated, and further, Ungar discloses encapsulation analysis as claimed. (Note rejection regarding claim 10)

**Regarding claim 23:**

The rejection of claim 19 is incorporated, and further, Ungar discloses the method being implemented in a Java environment as claimed. (Note rejection regarding claim 4)

**Regarding claim 24:**

The rejection of claim 19 is incorporated, and further, Ungar discloses identifying fields and objects as constants as claimed (Note rejection regarding claim 6)

**Regarding claim 25:**

The rejection of claim 19 is incorporated, and further, Ungar discloses identifying isolation faults as claimed (Note rejection regarding claim 5)

**Regarding claim 26:**

Ungar teaches:

- a device for detecting mutability of variables, objects, fields, and classes in a program component ("The invention detects whether the variable is type immutable..." in col. 7 lines 30-31)
- said component being written in an object oriented programming language ("Another embodiment optimizes OOP methods..." in col. 8 lines 49-50)
- a layer of at least one core library and at least one data-flow analysis engine ("the 'determine type usage pattern' procedure..." in col. 8 lines 30-31)
- a layer of at least one utility module, for using the results of the at least one data analysis engine to generate basic results ("an 'optimize access to variable' procedure optimizes the computer instructions used to access the data-values stored in the variable dependent on the results of the 'determine type usage pattern' procedure..." in col. 8 lines 42-45)
- a layer of at least one mutability sub-analysis module, for generating final results ("The invention detects variables that have immutable types (from the 'determine type usage pattern procedure)..." in col. 8 lines 61-62)
- wherein a variable is mutable if its state ever changes after said variable is initialized, the state of said variable being its value together with a state of any referenced object ("Mutable type variables store data-values of different types" in col. 5 lines 8-9)
- wherein an object is mutable if its state ever changes after said object is initialized, said state of said object being a set of states of all associated variables ("Mutable type variables store data-values of different types" in col. 5 lines 8-9. Further, "the 'z instance variable' storage contains the instance variables for the 'z object' structure" in col. 7 lines 7-8)

- wherein a field is mutable if any variable corresponding to said field is mutable ("Mutable type variables store data-values of different types" in col. 5 lines 8-9. Further, fields are inherently an object of the system.)
- wherein a class is mutable if any instance fields implemented by said class are mutable ("Mutable type variables store data-values of different types" in col. 5 lines 8-9. Further, A 'z object' structure contains information that is specific to each instantiation of an object of the 'z' class/map" in col. 7 lines 1-3)

as claimed.

**Regarding claim 27:**

The rejection of claim 26 is incorporated, and further, Ungar discloses a library for collecting and manipulating static information about the program component as claimed ("constructing a plurality of class/map structures associated with said variable, at least one of said plurality of class/map structures being dependent on said type usage pattern…" in col. 14 lines 26-29.)

**Regarding claim 28:**

The rejection of claim 26 is incorporated, and further, Ungar discloses a library for allowing a user to read classfiles as claimed ("constructing a plurality of class/map structures associated with said variable, at least one of said plurality of class/map structures being dependent on said type usage pattern…" in col. 14 lines 26-29.)

**Regarding claim 29:**

The rejection of claim 26 is incorporated, and further, Ungar discloses an intra-procedural data analysis engine for iteratively computing an effect of an instruction on information as claimed ("the 'determine type usage pattern' procedure determines whether only data-values having a specific type are stored in the variable" in col. 8 lines 30-32)

**Regarding claim 30:**

The rejection of claim 26 is incorporated, and further, Ungar discloses an intra-procedural data analysis engine for computing the effect of a method on information as claimed ("the 'determine type usage pattern' procedure determines whether only data-values having a specific type are stored in the variable" in col. 8 lines 30-32)

**Regarding claim 31:**

The rejection of claim 26 is incorporated, and further, Ungar discloses a type analysis utility module for identifying a set of possible types for each instruction as claimed ("The 'maintain type identifier' procedure saves the type of a data-value stored in a variable in a type identifier associated with the variable" in col. 8 lines 19-21)

**Regarding claim 32:**

The rejection of claim 26 is incorporated, and further, Ungar discloses a reachability analysis utility module as claimed ("The 'maintain type identifier' procedure saves the type of a data-value stored in a variable in a type identifier associated with the variable" in col. 8 lines 19-21)

**Regarding claim 33:**

The rejection of claim 26 is incorporated, and further, Ungar discloses a value modification mutability sub-analysis module as claimed ("the 'determine type usage pattern' procedure determines whether only data-values having a specific type are stored in the variable" in col. 8 lines 30-32)

**Regarding claim 34:**

The rejection of claim 26 is incorporated, and further, Ungar discloses an object modification mutability sub-analysis module as claimed ("the 'determine type usage pattern' procedure determines whether only data-values having a specific type are stored in the variable" in col. 8 lines 30-32)

**Regarding claim 35:**

The rejection of claim 26 is incorporated, and further, Ungar discloses a variable accessibility mutability sub-analysis module as claimed ("the 'determine type usage pattern' procedure determines whether only data-values having a specific type are stored in the variable" in col. 8 lines 30-32. Further, "if the 'mutable type variable' decision procedure determines that the passed entities are type immutable..." in col. 10 lines 51-53)

**Regarding claim 36:**

The rejection of claim 26 is incorporated, and further, Ungar discloses an object accessibility mutability sub-analysis module as claimed ("the 'determine type usage pattern' procedure determines whether only data-values having a specific type are stored in the variable" in col. 8 lines 30-32)

**Regarding claim 37:**

The rejection of claim 26 is incorporated, and further, Ungar discloses a breakage of encapsulation mutability sub-analysis module as claimed ("the 'determine type usage pattern' procedure determines whether only data-values having a specific type are stored in the variable" in col. 8 lines 30-32)

**Regarding claims 38-71:**

Claims 38-59 recite a computer system for performing the methods of claims 1-25 and are rejected for the reasons set forth in connection with claims 1-25, and further, claims 60-71 recite a computer system performing the same steps of the device in claims 26-37 and are rejected for the reasons set forth in connection with claims 26-37.

## *Conclusion*

9. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

10. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Trent J Roche whose telephone number is (703)305-4627. The examiner can normally be reached on Monday - Friday, 9:00 am - 6:30 pm.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Kakali Chaki can be reached on (703)305-9662. The fax phone number for the organization where this application or proceeding is assigned is (703) 872-9306.

Any inquiry of a general nature or relating to the status of this application or proceeding should be directed to the receptionist whose telephone number is (703)305-3900.

[illegible]

Trent J Roche
Examiner

**ANTONY NGUYEN-BA**
**PRIMARY EXAMINER**

Art Unit 2124

TJR